

# Richer Convolutional Features for Edge Detection

Yun Liu   Ming-Ming Cheng   Xiaowei Hu   Kai Wang  
CCCE, Nankai University

Xiang Bai  
HUST

<https://mmcheng.net/rcfEdge/>

## Abstract

In this paper, we propose an accurate edge detector using richer convolutional features (RCF). Since objects in nature images have various scales and aspect ratios, the automatically learned rich hierarchical representations by CNNs are very critical and effective to detect edges and object boundaries. And the convolutional features gradually become coarser with receptive fields increasing. Based on these observations, our proposed network architecture makes full use of multiscale and multi-level information to perform the image-to-image edge prediction by combining all of the useful convolutional features into a holistic framework. It is the first attempt to adopt such rich convolutional features in computer vision tasks. Using VGG16 network, we achieve state-of-the-art results on several available datasets. When evaluating on the well-known BSDS500 benchmark, we achieve ODS F-measure of **.811** while retaining a fast speed (**8 FPS**). Besides, our fast version of RCF achieves ODS F-measure of **.806** with **30 FPS**.

## 1. Introduction

Edge detection, which aims to extract visually salient edges and object boundaries from natural images, has retained one of the main challenges in computer vision for several decades. It is usually considered as a low-level technique, and varieties of high-level tasks have greatly benefited from the development of edge detection, such as object detection [15, 48], object proposal [47, 52] and image segmentation [1, 3, 8].

Typically, traditional methods first extract local cues of brightness, colors, gradients and textures, or other manual designed features like Pb [33], gPb [2], and Sketch tokens [29], then sophisticated learning paradigms [12, 49] are used to classify edge and non-edge pixels. Although edge detection approaches using low-level features have made great improvement in these years, their limitations are also obvious. For example, edges and boundaries are often defined to be semantically meaningful, however, low-level cues are

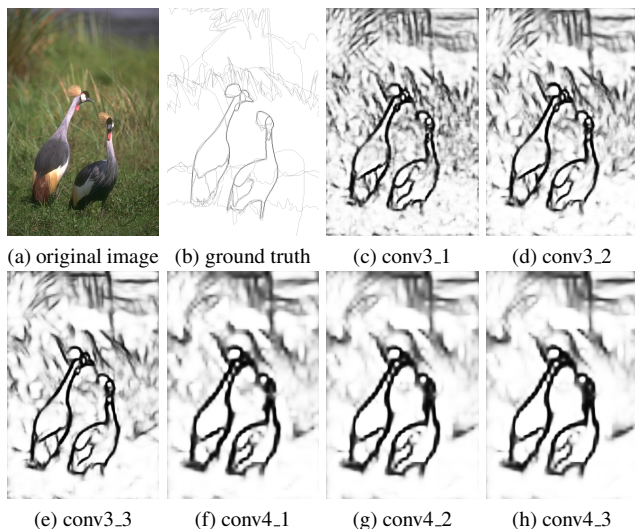


Figure 1: We build a simple network based on VGG16 [43] to obtain side outputs of conv3\_1, conv3\_2, conv3\_3, conv4\_1, conv4\_2 and conv4\_3. One can clearly see that convolutional features become coarser gradually, and the intermediate layers conv3\_1, conv3\_2, conv4\_1, and conv4\_2 contain lots of useful fine details that do not appear in other layers.

certainly difficult to represent object-level information. Under these circumstances, gPb [2] and Structured Edges [12] try to use complex strategies to capture global features as much as possible, but global information extracted in these ways is tiny.

In the last several years, convolutional neural networks (CNNs) have led fashion in the vision community by substantially advancing the state-of-the-art of various tasks, including image classification [27, 43, 45], object detection [18, 19, 36] and semantic segmentation [7, 31] etc. Since CNNs have strong capability to learn high-level representations of natural images automatically, there is a trend of using convolutional networks to perform edge detection recently. Some well-known CNN methods have pushed forward this field significantly, such as DeepEdge [4], N<sup>4</sup>-Fields [17], CSCNN [23], DeepContour [40], and HED

[50]. Our algorithm falls into this category as well.

To see the effectiveness of different convolution layers in edge detection, we build a simple network to obtain side outputs of intermediate layers using VGG16 [43] which has five convolution stages. Fig. 1 shows an example. We find that convolutional features become coarser gradually and intermediate layers contain lots of useful fine details. On the other hand, we know that richer convolutional features are important for many vision tasks, so many researchers try to develop deeper networks. However, the networks will be hard to converge when going deeper because of vanishing/exploding gradients and training data shortage. So why don't we make full use of the CNN features we have now? Our motivation is based on these observations. Unlike previous CNN methods, our novel network makes full use of multi-level CNN features to make the pixel-wise prediction in an image-to-image fashion, and thus could obtain accurate representations for objects or object parts in different scales. Concretely speaking, we make a bold attempt to utilize the CNN features from all the convolution layers in a unified framework that can be easily generalized to other vision tasks. By carefully designing a universal strategy to combine hierarchical CNN features, our system performs very well in edge detection.

When evaluating on BSDS500 dataset, we achieve state-of-the-art results with ODS F-measure of **.811**, which is higher than human eye (ODS F-measure 0.800). Our detector is also very efficient, achieving **8 FPS**. The fast version of RCF achieves ODS F-measure of **.806** with **30 FPS**.

## 2. Related Works

Since edge detection was set as one of the most fundamental problems in computer vision [13, 16, 39], researchers have struggled on it for nearly 50 years, and there have emerged a large number of materials. Broadly speaking, we can roughly categorize these approaches into three groups: early pioneering ones, manually designed features and learning based ones and deep learning based ones. Here we briefly review some important issues happened in the past few decades.

Early pioneering methods mainly focused on the utilization of intensity and color gradients. Robinson [39] discussed a quantitative measure in choosing color coordinates for the extraction of visually significant edges and boundaries. [32, 46] presented zero-crossing theory based algorithms. Sobel [44] proposed the famous Sobel operator to compute the gradient map of an image, and then obtain edges by thresholding the gradient map. An extended version of Sobel, named Canny [6], added Gaussian smooth as a preprocessing step and used bi-threshold to get edges. In this way, Canny is more robust to noise. And it is still very popular across various tasks now because of its notable efficiency. However, these early methods seem to have poor

accuracy and thus are difficult to adapt to today's applications.

Later, researchers tended to manually design features using low-level cues intensity, gradient, and texture, and then employ sophisticated learning paradigm to classify edge and non-edge pixels [11, 37]. Konishi *et al.* [26] proposed the first data-driven methods by learning the probability distributions of responses that correspond to two sets of edge filters. Martin *et al.* [33] formulated changes in brightness, color, and texture as Pb feature, and trained a classifier to combine the information from these features. Arbeláez *et al.* [2] developed Pb into gPb by using standard Normalized Cuts [41] to combine above local cues into a globalization framework. Lim [29] proposed novel features, Sketch tokens that can be used to represent the mid-level information. Dollár *et al.* [12] employed random decision forests to represent the structure presented in local image patches. Inputting color and gradient features, the structured forests output high-quality edges. Although above edge detectors have made brilliant achievements, development towards this direction is very slow, especially in recent years. The main reason is that using non-deep learning models to represent object-level information is a big challenge.

With the vigorous development of deep learning recently, a series of deep learning based approaches have been invented. Ganin *et al.* [17] proposed N<sup>4</sup>-Fields that combines CNNs with the nearest neighbor search. Shen *et al.* [40] partitioned contour data into subclasses and fit each subclass by learning model parameters. Hwang *et al.* [23] considered contour detection as a per-pixel classification problem. They employed DenseNet [24] to extract a feature vector for each pixel, and then SVM classifier is used to classify each pixel into the edge or non-edge class. Xie *et al.* [50] recently developed an efficient and accurate edge detector, HED which performs image-to-image training and prediction. This holistically-nested architecture connects their side output layers, which is composed of one convolution layer with kernel size 1, one deconvolution layer and one softmax layer, to the last convolution layer of each stage in VGG16 [43]. More recently, Liu *et al.* [30] used relaxed label generated by bottom-up edges to guide the training process of HED, and achieved a little bit of improvement. Li *et al.* [28] proposed a complex model for unsupervised learning of edge detection, but the performance is worse than training on the limited BSDS500 dataset.

Above CNNs advance the state-of-the-art significantly in a short span of two years, but all of them can not make full use of CNN features that represent different object-level information respectively. In this paper, we propose a fully convolutional network to combine features from each CNN layer efficiently. As far as we know, our novel network architecture is the first attempt to use all convolution layers in this community. We will detail our method below.

### 3. Richer Convolutional Features (RCF)

#### 3.1. Network Architecture

Inspired by previous literature in deep learning, we design our network by modifying VGG16 network [43]. VGG16 network that composes of 13 convolution layers and 3 fully connected layers has achieved state-of-the-art in a variety of tasks, such as image classification [43] and ob-

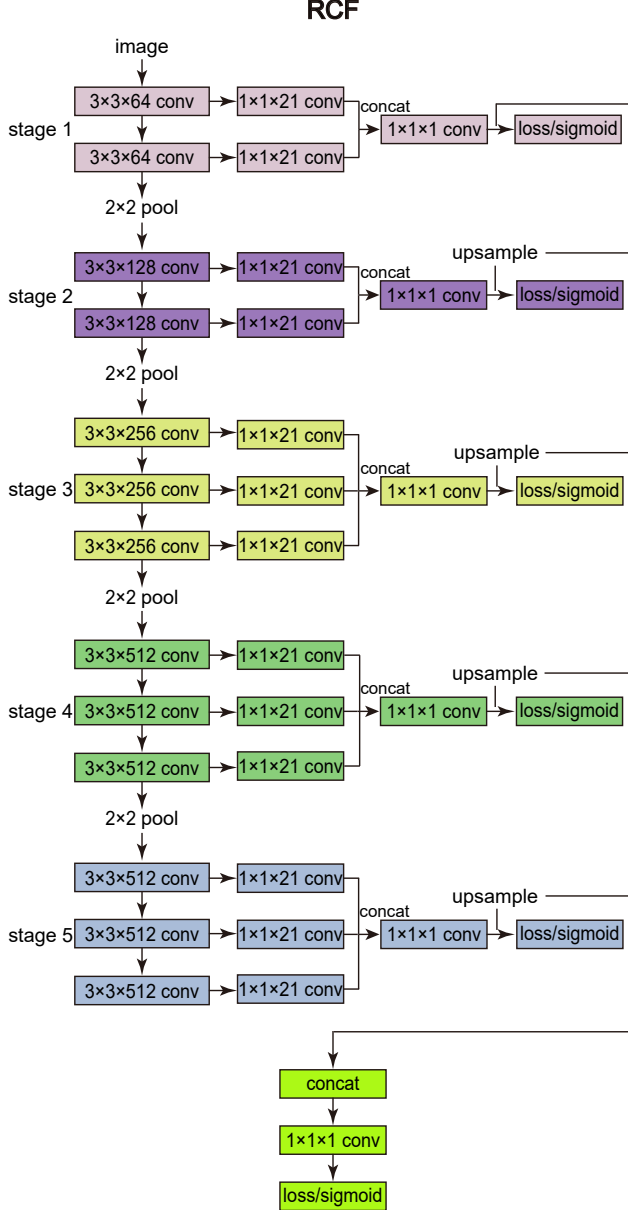


Figure 2: Our RCF network architecture. The input is an image with arbitrary sizes, and our network outputs an edge possibility map in the same size.

ject detection [18, 19, 36] *etc.* Its convolution layers are divided into five stages, in which a pooling layer is connected after each stage. The useful information obtained by each convolution layer becomes coarser with its receptive field size increasing. Detailed receptive field sizes of different layers can be seen in Tab. 1. Making full use of this rich hierarchical information will help a lot. The motivation of our network design lies here.

Table 1: Detailed receptive field and stride sizes of standard VGG16 net [43].

layer	conv1_1	conv1_2	pool1	conv2_1	conv2_2	pool2
rf size	3	5	6	10	14	16
stride	1	1	2	2	2	4
layer	conv3_1	conv3_2	conv3_3	pool3	conv4_1	conv4_2
rf size	24	32	40	44	60	76
stride	4	4	4	8	8	8
layer	conv4_3	pool4	conv5_1	conv5_2	conv5_3	pool5
rf size	92	100	132	164	196	212
stride	8	16	16	16	16	32

The novel network proposed by us is shown in Fig. 2. Compared with VGG16, our modifications can be described as following:

- We cut all the fully connected layers and the pool5 layer. On the one side, because fully connected layers are computationally expensive, we trim them to save executable time and memory. On the other hand, adding pool5 layer will increase the stride by two times, and it's harmful for edge localization.
- Each convolution layer in VGG16 is connected to a convolution layer with kernel size  $1 \times 1$  and channel depth 21. And the resulting layers in each stage are concatenated to obtain hybrid features.
- An  $1 \times 1 \times 1$  convolution layer follows each concat layer. Then, a deconvolution layer is used to up-sample this feature map.
- A cross-entropy loss/sigmoid layer is connected to the up-sampling layer in each stage.
- All the up-sampling layers are concatenated. Then an  $1 \times 1$  convolution layer and a cross-entropy loss /sigmoid layer are followed.

Hence, we combine hierarchical features from all the convolution layers into a holistic framework, in which all joint parameters are learned automatically. Since receptive field sizes of convolution layers in VGG16 are different from each other, our network can learn multiscale, including low-level and object-level, information that is helpful to



Figure 3: Several examples of the outputs in each stage of RCF. The top line is the original images from BSDS500 [2]. From second to sixth line is the output of stage 1, stage 2 and stage 3, stage 4 and stage 5 respectively.

edge detection. We show the intermediate results from each stage in Fig. 3. From top to bottom, the edge response becomes coarser while obtaining strong response at the larger object or object part boundaries. It is consistent with our expectation, in which convolution layers will learn to detect the larger objects with the receptive field size increasing. Since our RCF model combines all the accessible convolution layers to employ richer features, it is expected to achieve a boost in accuracy.

In order to investigate whether including additional non-linearity helps, we connecting ReLU layer after  $1 \times 1 \times 21$  or  $1 \times 1 \times 1$  convolution layers in each stage. However, the network performs worse. Especially, when we attempt to add nonlinear layers to  $1 \times 1 \times 1$  convolution layers, the network can not converge properly.

### 3.2. Improved Loss Function

Edge datasets in this community are usually labeled by several annotators using their knowledge about the presences of objects and object parts. Though humans vary in cognition, these human-labeled edges for the same image share high consistency. For each image, we average all the ground truth to generate an edge probability map, which ranges from 0 to 1. 0 means no annotator labeled at this pixel, and 1 means all annotators have labeled at this pixel. We consider the pixels with edge probability higher than  $\eta$  as positive samples and the pixels with edge proba-

bility equal to 0 as negative samples. Otherwise, if a pixel is marked by fewer than  $\eta$  of the annotators, this pixel may be semantically controversial to be an edge point. Thus, whether regarding it as positive or negative samples will confuse networks. So we ignore pixels in this category.

We compute the loss at every pixel with respect to pixel label as

$$l(X_i; W) = \begin{cases} \alpha \cdot \log(1 - P(X_i; W)) & \text{if } y_i = 0 \\ 0 & \text{if } 0 < y_i \leq \eta \\ \beta \cdot \log P(X_i; W) & \text{otherwise,} \end{cases} \quad (1)$$

in which

$$\begin{aligned} \alpha &= \lambda \cdot \frac{|Y^+|}{|Y^+| + |Y^-|} \\ \beta &= \frac{|Y^-|}{|Y^+| + |Y^-|}. \end{aligned} \quad (2)$$

$Y^+$  and  $Y^-$  denote positive sample set and negative sample set respectively.  $\lambda$  is a hyper-parameter to balance positive and negative samples.  $X_i$  and  $y_i$  are activation value (CNN feature vector) and ground truth edge probability at pixel  $i$ , respectively.  $P(X)$  is the standard sigmoid function. And  $W$  denotes all the parameters that will be learned in our architecture. Therefore, our novel loss function can be formulated as

$$L(W) = \sum_{i=1}^{|I|} \left( \sum_{k=1}^K l(X_i^{(k)}; W) + l(X_i^{fuse}; W) \right), \quad (3)$$

where  $X_i^{(k)}$  is the activation value from stage  $k$  while  $X_i^{fuse}$  is from fusion layer.  $|I|$  is the number of pixels in image  $I$ , and  $K$  is the number of stages (equals to 5 here).

### 3.3. Multiscale Hierarchical Edge Detection

In single scale edge detection, we input an original image into our fine-tuned RCF network, then, the output is an edge probability map. To further explore the effectiveness of multiscale hierarchies, we use image pyramids at the test. Concretely speaking, we resize an image to construct an image pyramid, and each of these images is input to our single-scale detector separately. Then, all resulting edge probability maps are resized to original image size using bilinear interpolation. At last, these maps are averaged to get a final prediction map. Fig. 4 shows a visualized pipeline of our multiscale algorithm. We also try to use weighted sum, but we find the simple average works very well. Considering the trade-off between accuracy and speed, we use three scales 0.5, 1.0, and 1.5 in this paper. When evaluating on BSDS500 [2] dataset, this simple multiscale strategy improves the ODS F-measure from 0.806 to 0.811. See Sec. 4 for details.



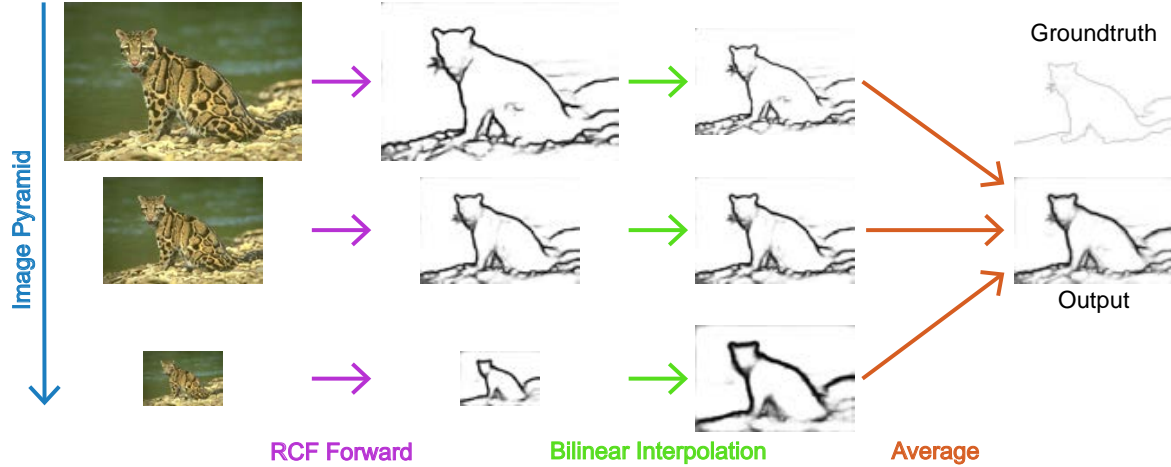


Figure 4: The pipeline of our multiscale algorithm. The original image is resized to construct an image pyramid. And these multiscale images are input to RCF network for a forward pass. Then, we use bilinear interpolation to restore resulting edge response maps to original sizes. A simple average of these edge maps will output high quality edges.

### 3.4. Comparison With HED

The most obvious difference between our RCF and HED [50] is in three parts. First, HED only considers the last convolution layer in each stage of VGG16, in which lots of helpful information to edge detection is missed. In contrast to it, RCF uses richer features from all the convolution layers, thus it can capture more object or object part boundaries accurately across a larger range of scales. Second, a novel loss function is proposed to treat training examples properly. We only consider the edge pixels that most annotators labeled as positive samples, since these edges are highly consistent and thus easy to train. Besides, we ignore edge pixels that are marked by a few annotators because of their confusing attributes. Thirdly, a simple way to obtain multiscale hierarchy is proposed to enhance edges.

Therefore, RCF is not only a reasonably extended version of HED, but also a bold attempt for edge detection in many ways. As far as I am concerned, it is the first framework to combine all the convolution layers in this community. Our evaluation results demonstrate the effectiveness (1.4% improvement in ODS F-measure over HED) of these choices. See Sec. 4 for details.

## 4. Experiment

We implement our network using the publicly available Caffe [25] which is well-known in this community. The VGG16 model that is pre-trained on ImageNet [10] is used to initialize our network. We change the stride of pool4 layer to 1 and use the atrous algorithm to fill the holes. In RCF training, the weights of  $1 \times 1 \times 1$  convolution layer in each stage are initialized from zero-mean Gaussian distributions with standard deviation 0.01 and biases are ini-

tialized to 0. The fusion layer weights are initialized to 0.2. Stochastic gradient descent (SGD) minibatch samples 1 image randomly in each iteration, and the iter size is set to 10. For other SGD hyper-parameters, the global learning rate is set depending on different dataset and will be divided by 10 after every 10k iterations. The momentum and weight decay are set to 0.9 and 0.0002 respectively. We run SGD for 40k iterations totally. The parameters  $\eta$  and  $\lambda$  in loss function are also set depending on training data. All experiments in this paper are finished using a NVIDIA TITAN X GPU.

Given an edge probability map, a threshold is needed to obtain the edge image. There are two choices to set this threshold. The first one is referred as optimal dataset scale (ODS) which employs a fixed threshold for all images in the dataset. And the second is called optimal image scale (OIS) which selects an optimal threshold for each image. We use F-measure ( $\frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$ ) of both ODS and OIS in our experiments.

### 4.1. BSDS500 Dataset

BSDS500 [2] is a widely used dataset in edge detection. It is composed of 200 training, 100 validation and 200 test images, and each image is labeled by 4 to 9 annotators. We utilize the training and validation sets for fine-tuning, and test set for evaluation. Data augmentation is the same as [50]. Inspired by RDS [30] and CEDN [51], we mix augmentation data of BSDS500 with flipped PASCAL VOC Context dataset [35] as training data. When training, we set loss parameters  $\eta$  and  $\lambda$  to 0.5 and 1.1, respectively. And initial learning rate is set to  $1e-6$ . When evaluating, standard non-maximum suppression (NMS) [12] is applied to thin detected edges. We compare our method with some

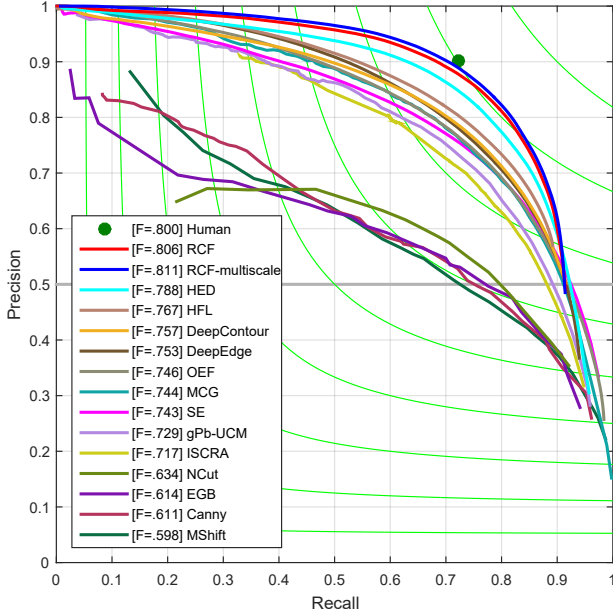


Figure 5: The evaluation results on standard BSDS500 [2] dataset. Both single-scale and multiscale versions of RCF achieve better performance than the human eye.

non-deep-learning algorithms, including Canny [6], EGB [14], gPb-UCM [2], IS CRA [38], MCG [3], MShift [9], NCut [41], SE [12], and OEF [22], and some recent deep learning based approaches, including DeepContour [40], DeepEdge [4], HED [50], HFL [5] and *etc.*

Fig. 5 shows the evaluation results. The performance of human eye in edge detection is known as .800 ODS F-measure. Both single-scale and multiscale versions of RCF achieve better results than human eye. When comparing with HED [50], ODS F-measures of our multiscale and single-scale RCF are 2.3% and 1.8% higher than it, respectively. And the precision-recall curves of our methods are also higher than HED’s. These significant improvements demonstrate the effectiveness of our richer convolutional features. All the convolution layers contain helpful hierarchical information, not only the last one in each convolution stage.

From single-scale RCF to multiscale version, the accuracy also has some improvement. It proves the effectiveness of our multiscale strategy. We also observe an interesting phenomenon in which the RCF curves are not as long as other methods when evaluated using the default parameters in BSDS500 benchmark. It may suggest that RCF tends only to remain very confident edges. However, it doesn’t matter to us, since the best ODS F-measure is much more important in practice. If we consider more pixels as edges, the recall will be higher, but it is meaningless. Others, one can find that all the deep learning based approaches outper-

Table 2: The comparison with some competitors on BSDS500 [2] dataset. †means GPU time.

Method	ODS	OIS	FPS
Canny [6]	.611	.676	28
EGB [14]	.614	.658	10
MShift [9]	.598	.645	1/5
gPb-UCM [2]	.729	.755	1/240
Sketch Tokens [29]	.727	.746	1
MCG [3]	.744	.777	1/18
SE [12]	.743	.763	2.5
OEF [22]	.746	.770	2/3
DeepContour [40]	.757	.776	1/30†
DeepEdge [4]	.753	.772	1/1000†
HFL [5]	.767	.788	5/6†
N <sup>4</sup> -Fields [17]	.753	.769	1/6†
HED [50]	.788	.808	<b>30</b> †
RDS [30]	.792	.810	<b>30</b> †
CEDN [51]	.788	.804	10†
RCF	.806	.823	<b>30</b> †
RCF-multiscale	<b>.811</b>	<b>.830</b>	<b>8</b> †

form non-deep ones. We think it is because CNNs can learn rich object-level information automatically, and this automatically learned information is more reliable than human-designed features. If more training data is accessible, the gap between deep and non-deep methods will be bigger. Others, NCut, EGB, Canny and MShift seem to perform worse than nowadays methods.

We show statistic comparison in Tab. 2. RCF substantially achieves the best performance. This fact demonstrates the effectiveness of our novel network architecture. RCF is also one of the fastest edge detectors. The single-scale RCF achieves **30** FPS, and the multiscale RCF can also achieve **8** FPS. Note that we only add some  $1 \times 1$  convolution layers to HED, so the time consumption is almost same as HED. Comparing with other deep learning based methods, like DeepEdge [4] and DeepContour [40], RCF is hundreds of times faster than them. Thus, we achieve the state-of-the-art in both accuracy and efficiency. And this speed of 30 FPS makes it easy to be applied in various tasks. If higher accuracy is needed, the multiscale version which is slightly slower is also a good choice.

## 4.2. Network Discussion

To further explore the effectiveness of our network architecture, we implement some thought networks using VGG16 [43] by connecting our richer feature side outputs to some convolution stages while connecting side outputs of HED to the other stages. With training only on BSDS500 [2] dataset, evaluation results of these thought networks are shown in Tab. 3. The last two lines of this table corre-

spond to HED and RCF, respectively. We can observe that all of these thought networks perform better than HED and worse than RCF that is fully connected to RCF side outputs. It clearly demonstrates the importance of our strategy of richer convolutional features.

Table 3: Results of some thought networks.

RCF Stage	HED Stage	ODS	OIS
1, 2	3, 4, 5	.792	.810
2, 4	1, 3, 5	.795	.812
4, 5	1, 2, 3	.790	.810
1, 3, 5	2, 4	.794	.810
3, 4, 5	1, 2	.796	.812
–	1, 2, 3, 4, 5	.788	.808
1, 2, 3, 4, 5	–	.798	.815

### 4.3. NYUD Dataset

NYUD [42] dataset is composed of 1449 densely labeled pairs of aligned RGB and depth images. Recently many works have conducted edge evaluation on it, such as [12, 49]. Gupta *et al.* [20] split NYUD dataset into 381 training, 414 validation and 654 testing images. We follow their settings and train our RCF network using training and validation sets in full resolution as in HED [50].

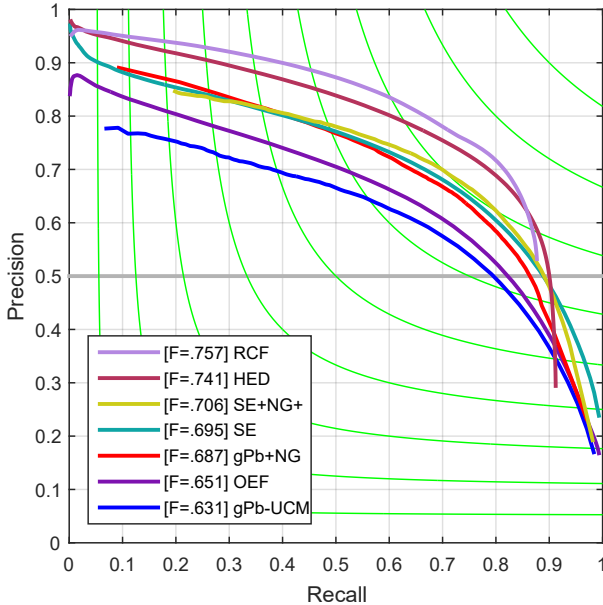


Figure 6: The evaluation results on NYUD [42] dataset. RCF is referred to single-scale version here.

We utilize depth information as HED [50] by using HHA [21], in which depth information is encoded into three channels: horizontal disparity, height above ground, and angle

with gravity. Thus HHA features can be represented as a color image. Then, two models for RGB images and HHA feature images are trained separately. In the training process,  $\lambda$  is set to 1.2, and global learning rates for RGB and HHA are both set to  $1e-6$ . Since NYUD only has one ground truth for each image,  $\eta$  is useless here. Other network settings are the same as used for BSDS500. At testing, the final edge predictions are defined by averaging the outputs of RGB model and HHA model. When evaluating, we increase localization tolerance, which controls the maximum allowed distance in matches between predicted edges and ground truth, from 0.0075 to 0.011.

Table 4: The comparison with some competitors on NYUD dataset [42]. †means GPU time.

Method	ODS	OIS	FPS
OEF [22]	.651	.667	1/2
gPb-UCM [2]	.631	.661	1/360
gPb+NG [20]	.687	.716	1/375
SE [12]	.695	.708	5
SE+NG+ [21]	.706	.734	1/15
HED-HHA [50]	.681	.695	<b>20</b> †
HED-RGB [50]	.717	.732	<b>20</b> †
HED-RGB-HHA [50]	.741	.757	10†
RCF-HHA	.705	.715	<b>20</b> †
RCF-RGB	.729	.742	<b>20</b> †
RCF-HHA-RGB	<b>.757</b>	<b>.771</b>	10†

We only compare our single-scale version of RCF with some famous competitors. OEF [22] and gPb-UCM [2] only use RGB images, while other methods employ both depth and RGB information. The precision-recall curves are shown in Fig. 6. RCF achieves the best performance on NYUD dataset, and the second place is HED [50]. Tab. 4 shows statistical comparison. We can see that RCF achieves better results than HED not only on separate HHA or RGB data, but also on the merged HHA-RGB data. For HHA and RGB data, ODS F-measure of RCF is 2.4% and 1.2% higher than HED, respectively. For merging HHA-RGB data, RCF is 1.6% higher than HED. Others, HHA edges performance worse than RGB, but averaging HHA and RGB edges achieving much higher results. It suggests that combining different types of information is very useful for edge detection, and it's also the reason why OEF and gPb-UCM perform worse than other methods.

### 4.4. Multicue Dataset

Recently, Multicue dataset is proposed by Mély *et al.* [34] to study psychophysics theory for boundary detection. It is composed of short binocular video sequences of 100 challenging natural scenes captured by a stereo camera. Each scene contains a left and right view short (10-

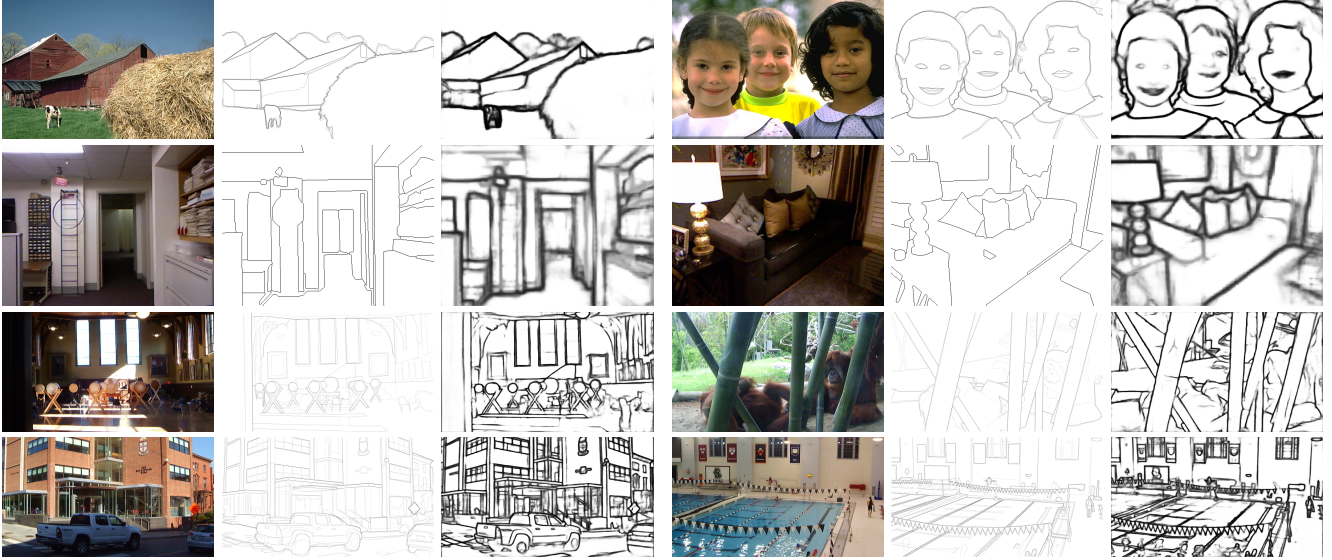


Figure 7: Some examples of RCF. From top to bottom: BSDS500 [2], NYUD [42], Multicue-Boundary [34], and Multicue-Edge [34]. From left to right: origin image, ground truth, RCF edge map, origin image, ground truth, and RCF edge map.

frame) color sequence. The last frame of the left image for each scene is labeled for two annotations, object boundaries and low-level edges. Unlike people who usually use boundary and edge interchangeably, they strictly defined boundary and edge according to visual perception at different stages. Thus, boundaries are referred to the boundary pixels of meaningful objects, and edges are abrupt pixels at which the luminance, color or stereo change sharply. In this subsection, we use boundary and edge as defined by Mély *et al.* [34] while considering boundary and edge having the same meaning in previous sections. As done in [34] and recent version of HED<sup>1</sup>, we randomly split these human-labeled images into 80 training and 20 test samples, and average the scores of three independent trials as final results. When training on Multicue,  $\lambda$  and initial learning rate are set to 1.1 and  $1e-6$  respectively. And  $\eta$  is set to 0.4 for boundary task and 0.3 for edge task. Since the image resolution of Multicue is very high, we randomly crop  $400 \times 400$  patches. When evaluating, we adjust the localization tolerance of benchmark [2] to 0.011 as on NYUD dataset.

We show evaluation results in Tab. 5. Our proposed RCF achieve substantially higher results than HED. For boundary task, RCF is 2.6% ODS F-measure higher and 2.8% OIS F-measure higher than HED. For edge task, RCF is 4.5% ODS F-measure higher and 5.1% OIS F-measure higher than HED. The great improvement on Multicue edge task may be due to more fine information extracted by RCF. Thus, we can infer that RCF can obtain not only more semantic features but also more fine detail features. This char-

Table 5: The comparison with some competitors on Multicue dataset [34].

Method	ODS	OIS
Human-Boundary [34]	.760 (.017)	–
Multicue-Boundary [34]	.720 (.014)	–
HED-Boundary [50]	.821 (.021)	.826 (.012)
RCF-Boundary	<b>.847 (.007)</b>	<b>.854 (.007)</b>
Human-Edge [34]	.750 (.024)	–
Multicue-Edge [34]	.830 (.002)	–
HED-Edge [50]	.827 (.013)	.829 (.007)
RCF-Edge	<b>.872 (.006)</b>	<b>.880 (.005)</b>

acteristic is very useful in many vision tasks. Note that the fluctuation of RCF is also much smaller than HED, which suggests RCF is more robust over different kinds of images.

## 5. Conclusion

In this paper, we propose a novel CNN architecture, RCF, that makes full use of semantic and fine detail features to carry out edge detection. We carefully design it as an extensible architecture. The resulting RCF method can obtain high-quality edges very efficiently, and this makes it promising to be easily applied to various tasks. RCF architecture can be seen as a development direction of fully connected network, like FCN [31] and HED [50]. It would be interesting to explore the effectiveness of our network architecture in other hot topics, such as salient object detection and semantic segmentation. Source code will be published with this paper.

<sup>1</sup>Details can be seen in recent journal version of HED which is available at the authors' homepage (<http://pages.ucsd.edu/~ztu/Publication.htm>).



## References

- [1] P. Arbeláez, M. Maire, C. Fowlkes, and J. Malik. From contours to regions: An empirical evaluation. In *IEEE CVPR*, pages 2294–2301. IEEE, 2009. **1**
- [2] P. Arbeláez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *IEEE TPAMI*, 33(5):898–916, 2011. **1, 2, 4, 5, 6, 7, 8**
- [3] P. Arbeláez, J. Pont-Tuset, J. T. Barron, F. Marques, and J. Malik. Multiscale combinatorial grouping. In *IEEE CVPR*, pages 328–335, 2014. **1, 6**
- [4] G. Bertasius, J. Shi, and L. Torresani. DeepEdge: A multi-scale bifurcated deep network for top-down contour detection. In *IEEE CVPR*, pages 4380–4389, 2015. **1, 6**
- [5] G. Bertasius, J. Shi, and L. Torresani. High-for-low and low-for-high: Efficient boundary detection from deep object features and its applications to high-level vision. In *IEEE ICCV*, pages 504–512, 2015. **6**
- [6] J. Canny. A computational approach to edge detection. *IEEE TPAMI*, (6):679–698, 1986. **2, 6**
- [7] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. *arXiv preprint arXiv:1412.7062*, 2014. **1**
- [8] M.-M. Cheng, Y. Liu, Q. Hou, J. Bian, P. Torr, S.-M. Hu, and Z. Tu. Hfs: Hierarchical feature selection for efficient image segmentation. In *ECCV*, pages 867–882. Springer, 2016. **1**
- [9] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE TPAMI*, 24(5):603–619, 2002. **6**
- [10] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE CVPR*, pages 248–255. IEEE, 2009. **5**
- [11] P. Dollár, Z. Tu, and S. Belongie. Supervised learning of edges and object boundaries. In *IEEE CVPR*, volume 2, pages 1964–1971. IEEE, 2006. **2**
- [12] P. Dollár and C. L. Zitnick. Fast edge detection using structured forests. *IEEE TPAMI*, 37(8):1558–1570, 2015. **1, 2, 5, 6, 7**
- [13] R. O. Duda, P. E. Hart, et al. *Pattern classification and scene analysis*, volume 3. Wiley New York, 1973. **2**
- [14] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *IJCV*, 59(2):167–181, 2004. **6**
- [15] V. Ferrari, L. Fevrier, F. Jurie, and C. Schmid. Groups of adjacent contour segments for object detection. *IEEE TPAMI*, 30(1):36–51, 2008. **1**
- [16] J. R. Fram and E. S. Deutsch. On the quantitative evaluation of edge detection schemes and their comparison with human performance. *IEEE TOC*, 100(6):616–628, 1975. **2**
- [17] Y. Ganin and V. Lempitsky.  $N^4$ -Fields: Neural network nearest neighbor fields for image transforms. In *ACCV*, pages 536–551. Springer, 2014. **1, 2, 6**
- [18] R. Girshick. Fast R-CNN. In *IEEE ICCV*, pages 1440–1448, 2015. **1, 3**
- [19] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *IEEE CVPR*, pages 580–587, 2014. **1, 3**
- [20] S. Gupta, P. Arbelaez, and J. Malik. Perceptual organization and recognition of indoor scenes from rgb-d images. In *IEEE CVPR*, pages 564–571, 2013. **7**
- [21] S. Gupta, R. Girshick, P. Arbeláez, and J. Malik. Learning rich features from rgb-d images for object detection and segmentation. In *ECCV*, pages 345–360. Springer, 2014. **7**
- [22] S. Hallman and C. C. Fowlkes. Oriented edge forests for boundary detection. In *IEEE CVPR*, pages 1732–1740, 2015. **6, 7**
- [23] J.-J. Hwang and T.-L. Liu. Pixel-wise deep learning for contour detection. *arXiv preprint arXiv:1504.01989*, 2015. **1, 2**
- [24] F. Iandola, M. Moskewicz, S. Karayev, R. Girshick, T. Darrell, and K. Keutzer. Densenet: Implementing efficient convnet descriptor pyramids. *arXiv preprint arXiv:1404.1869*, 2014. **2**
- [25] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACM MM*, pages 675–678. ACM, 2014. **5**
- [26] S. Konishi, A. L. Yuille, J. M. Coughlan, and S. C. Zhu. Statistical edge detection: Learning and evaluating edge cues. *IEEE TPAMI*, 25(1):57–74, 2003. **2**
- [27] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1097–1105, 2012. **1**
- [28] Y. Li, M. Paluri, J. M. Rehg, and P. Dollár. Unsupervised learning of edges. In *IEEE CVPR*, pages 1619–1627, 2016. **2**
- [29] J. J. Lim, C. L. Zitnick, and P. Dollár. Sketch tokens: A learned mid-level representation for contour and object detection. In *IEEE CVPR*, pages 3158–3165, 2013. **1, 2, 6**
- [30] Y. Liu and M. S. Lew. Learning relaxed deep supervision for better edge detection. In *IEEE CVPR*, pages 231–240, 2016. **2, 5, 6**
- [31] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *IEEE CVPR*, pages 3431–3440, 2015. **1, 8**
- [32] D. Marr and E. Hildreth. Theory of edge detection. *Proceedings of the Royal Society of London B: Biological Sciences*, 207(1167):187–217, 1980. **2**
- [33] D. R. Martin, C. C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE TPAMI*, 26(5):530–549, 2004. **1, 2**
- [34] D. A. Mély, J. Kim, M. McGill, Y. Guo, and T. Serre. A systematic comparison between visual cues for boundary detection. *Vision research*, 120:93–107, 2016. **7, 8**
- [35] R. Mottaghi, X. Chen, X. Liu, N.-G. Cho, S.-W. Lee, S. Fidler, R. Urtasun, and A. Yuille. The role of context for object detection and semantic segmentation in the wild. In *IEEE CVPR*, pages 891–898, 2014. **5**
- [36] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*, pages 91–99, 2015. **1, 3**
- [37] X. Ren. Multi-scale improves boundary detection in natural images. In *ECCV*, pages 533–545. Springer, 2008. **2**

- [38] Z. Ren and G. Shakhnarovich. Image segmentation by cascaded region agglomeration. In *IEEE CVPR*, pages 2011–2018, 2013. 6
- [39] G. S. Robinson. Color edge detection. *Optical Engineering*, 16(5):165479–165479, 1977. 2
- [40] W. Shen, X. Wang, Y. Wang, X. Bai, and Z. Zhang. Deep-Contour: A deep convolutional feature learned by positive-sharing loss for contour detection. In *IEEE CVPR*, pages 3982–3991, 2015. 1, 2, 6
- [41] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE TPAMI*, 22(8):888–905, 2000. 2, 6
- [42] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgb-d images. In *European Conference on Computer Vision*, pages 746–760. Springer, 2012. 7, 8
- [43] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 1, 2, 3, 6
- [44] I. Sobel. Camera models and machine perception. Technical report, DTIC Document, 1970. 2
- [45] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *IEEE CVPR*, pages 1–9, 2015. 1
- [46] V. Torre and T. A. Poggio. On edge detection. *IEEE TPAMI*, (2):147–163, 1986. 2
- [47] J. R. Uijlings, K. E. van de Sande, T. Gevers, and A. W. Smeulders. Selective search for object recognition. *IJCV*, 104(2):154–171, 2013. 1
- [48] S. Ullman and R. Basri. Recognition by linear combinations of models. *IEEE TPAMI*, 13(10):992–1006, 1991. 1
- [49] R. Xiao-feng and L. Bo. Discriminatively trained sparse code gradients for contour detection. In *NIPS*, pages 584–592, 2012. 1, 7
- [50] S. Xie and Z. Tu. Holistically-nested edge detection. In *IEEE ICCV*, pages 1395–1403, 2015. 1, 2, 5, 6, 7, 8
- [51] J. Yang, B. Price, S. Cohen, H. Lee, and M.-H. Yang. Object contour detection with a fully convolutional encoder-decoder network. *arXiv preprint arXiv:1603.04530*, 2016. 5, 6
- [52] C. L. Zitnick and P. Dollár. Edge boxes: Locating object proposals from edges. In *ECCV*, pages 391–405. Springer, 2014. 1